# Advance DSA

## Arrays – 1 (7th Jan 2022)

1. **Kadane’s Algorithm**

Used to find the maximum sum of any contiguous array / sub-array. Important point to note is that the array elements should be continuous in nature.

Helps to traverse the array in O(N) TC and without using any extra auxiliary space.

Problems based

1. [**Important**] Given an array, find the maximum sum of contiguous elements. *[Amazon, Directi, Google, LinkedIn].*

Follow-up question is to return the start and end indexes along with the maximum sum. i.e. (start, end) and max.

1. Given an array of integers, A of length N, find out the maximum sum sub-array of non-negative numbers from A.

The sub-array should be contiguous i.e., a sub-array created by choosing the second and fourth element and skipping the third element is invalid.

1. **Marking the presence of the numbers using the same array [*Important – Revise*]**

Reason we are able to use the original array to mark the presence of the numbers is because we are given the range and we are only concerned about the numbers from [1, N] and we also know the size of the array is also N. Hence, we can use the indices to mark the elements presence. If the elements are negative or the range varies than this approach can not be used. We are able to achieve the O(N) TC and without even using any extra space like HashSet.

Problems based

1. Given an array A, find the smallest positive number that is not present in the array. *[Google].*
2. **Preprocessing Data**

Preprocess the given input data to achieve the final solution. It also helps to reduce the overall time complexity.

Problems based

1. Rainwater trapping problem.

Follow up question could you come up with a solution without using any extra auxiliary space? [*Facebook, Google, GS*]

## Graphs – DFS

Problems based

**200. Number of Islands**

We need to run the loops to traverse the entire matrix and whenever there is a 1 in the matrix (which represents an island) that means it will result in increase of island count, we need to make a call to markVisited/dfs and ensure that we mark this element as visited in the matrix in the dfs call and all other 4-dimensionally connected 1s as visited as well, dfs method here will not return anything.

**733. Flood Fill**

Here, we don’t need the for loops to start iterating over the entire matrix. Basically, we need to find the color of the starting element and check if it is same as new color, if yes return the original matrix array. Otherwise, make a dfs call… (matrix, srcRow, srcCol, srcColor, newColor)… Once, we are inside the method, just check for boundary condition and return… otherwise set the color of the given element to newColor… and then finally call the dfs method on 4-dimensionally connected elements.

**695. Max Area of Island**

We need to run the loops to traverse the entire matrix and whenever there is a 1 in the matrix (which represents part of land) that means it will help in the count of area of an island, we need to make a call to findArea/ dfs call that will return an integer representing the maximum area formed by this island…

In findArea, we will have the baseCondition that will return 0. We will mark the current element as visited by setting to 0.. and then set the areacount to 1.. and then add make the 4- dimensionally connected findArea calls and add to areaCount each time. And finally return the areaCount.

## Graphs – BFS

### Important Point

1. We will always use BFS to find shortest distance in the Graph. DFS will result in exponential time complexity due to multiple ways to reach between two nodes, while DFS will run in polynomial time complexity.

On the other hand, DFS could be used in Tree for finding minimum distance between two nodes as there would be only 1 possible shortest unique path.

Problems based

1. Check Same Binary Tree
2. Check Symmetric Binary Tree
3. Level Order Traversal (Done with help of queue)
4. BinaryTree Zigzag level order traversal (done with help of queue and a flag)
5. Minimum/maximum depth of binary tree